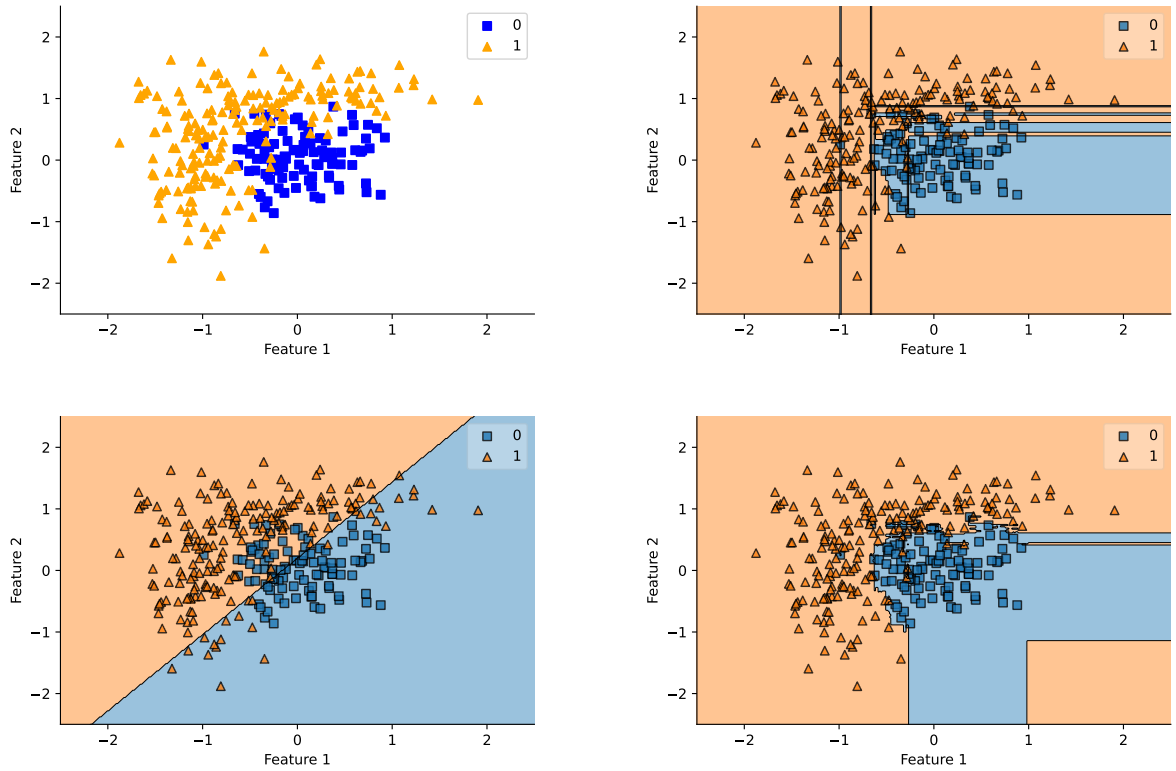
Data 102 (Fall 2022) **Names:**
Discussion Worksheet 7
10/14/2022

---

1. (a) The four images below show a classification data set, along with the decision boundary for each of three models: (1) logistic regression, (2) a decision tree with no depth limit, and (3) a random forest with no depth limit. Match each model to the corresponding image.



Solution: Clockwise: data; decision tree; logistic regression; random forest classifier. Random forest decision boundaries are generated by maintaining multiple decision trees, each with different decisions for splitting the data, and then averaging over the classification that each tree suggests to determine a final classification. Therefore, the decision boundary will be smoother for random forests, because outlier decision boundaries for one particular decision tree in the random forest will be averaged out. Logistic regression models have a linear decision boundary. You can see this by recalling that for logistic regression models,

$$\ln \frac{P(Y=1|X)}{P(Y=0|X)} = \beta_0 + \beta_1 X.$$

The decision boundary is the set of all X such that $P(Y=1|X) = P(Y=0|X)$. Therefore, for all X on the decision boundary, the equation above simplifies to $\beta_0 + \beta_1 X = 0$.

(b) Recall the bias-variance decomposition for the squared error:

$$E[(\hat{y}(x) - y)^2] = \underbrace{E\left[(\hat{y}(x) - E[\hat{y}(x)])^2\right]}_{\text{Variance of prediction } \hat{y}(x)} + \underbrace{(E[\hat{y}(x)] - y)^2}_{\text{Bias}^2 \text{ of } \hat{y}(x)}.$$

Circle the correct word for each sentence:

- The bias of logistic regession is (LARGER THAN / SMALLER THAN / EQUAL TO) that of the decision tree.
- The bias of the decision tree is (LARGER THAN / SMALLER THAN / EQUAL TO) that of the random forest model.
- The variance of the decision tree is (LARGER THAN / SMALLER THAN / EQUAL TO) that of the random forest model.

Solution:

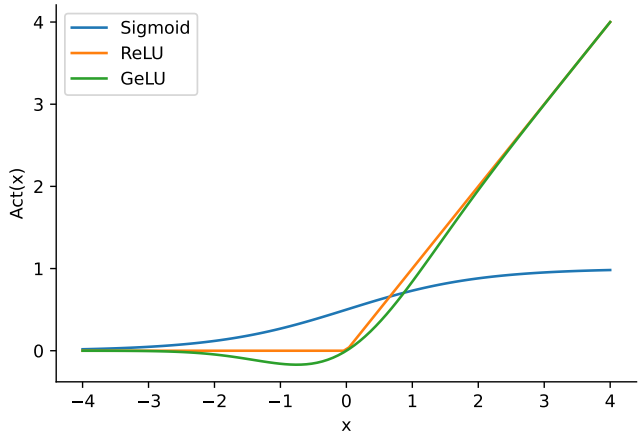
- Bias of logistic regression is LARGER THAN bias of a decision tree. A good general rule of thumb is that models which have more structural assumptions (e.g. logistic regression, which assumes that the log-odds is linear in the data X) have higher bias. They are not as flexible so may not do as well on the training data but on the flip side, since they are not over-fitted to the training data, they do better on unseen data which may be dissimilar to the training data. The opposite is true for variance – less structured models (like decision trees, which are very flexible to the training data) typically have higher variance. They fit the training data well, but may not perform well on new data which is dissimilar to the training data.
- Bias of decision tree is EQUAL TO that of a random forest model. Recall that bias is just an expectation (i.e. the expected difference between decision and reality). Since random forests are averages of decision trees, the expectation in the bias absorbs this averaging, and the bias of both models is the same.
- Variance of decision tree is LARGER THAN variance of the random forest model. Since random forests don't just use one splitting of the data to construct classification decisions, but average over multiple splittings, they are less over-fitted to the training data and generalize better, thus have lower variance.

2. We saw in lecture that neural networks can learn complex, non-linear patterns from data using activation functions. One simple activation function which you've seen in this class before is the *sigmoid* function used in Logistic Regression, defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x))}$$

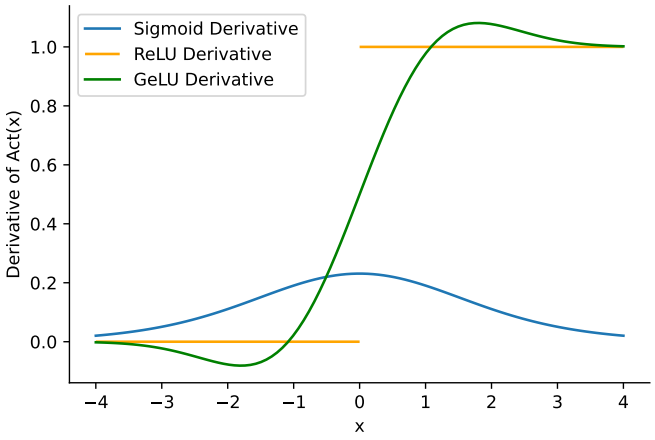
However, most modern-day implementations of neural networks avoid using sigmoid functions for the nonlinearity, and instead favor functions like the ReLU (*Rectified Linear Unit*) or GELU. The ReLU function is defined as $\text{ReLU}(x) = \max(0, x)$, and the GELU has a formula involving the CDF of a Gaussian (not given here).

We can visualize what these activation functions look like in the plot below:



(a) Referencing the plot above, sketch the derivatives of the sigmoid and ReLU activation functions, overlaid on the same plot.

Solution: The plot is given below:



Note that the derivative of ReLU is discontinuous at 0 since the original ReLU function has a jump at that point.

(b) What happens to the derivative for each of the activation functions as the input gets large?

Solution: For values of its input far away from 0 (positive or negative), the derivative of the sigmoid function approaches 0. In comparison, the derivatives of the ReLU and GeLU functions stay at or near 1 for positive values of x. The derivatives of ReLU and GeLU do not "decay" to 0 like the derivative of sigmoid for both large and small values of x.

(c) When optimizing neural networks, why is it bad for the gradient values to be (close to) 0? Why do ReLU and GeLU do a better job than sigmoid of avoiding this problem?

Solution: Since backpropagation involves multiplying many derivatives together, as soon as we have one zero value in our computation graph, the whole gradient will vanish to 0. This is problematic because when updating the network's weights, a gradient value of 0 means that the weight will not be changed; in other words, the network will

not learn! To avoid this problem, selecting an activation function which avoids vanishing gradients for at least large valuess of x, like ReLU or GeLU, can work. GeLU offers a compromise between the vanishing gradients problem of sigmoid and dead gradients problem of ReLU (which is 0 for all negative x). Some other methods to avoid the vanishing gradient problem (which aren't in scope for this class) include using more robust network architectures, batch normalization, and various techniques for weight initialization.

3. Consider a two-layer neural network that computes a real-valued function of the form

$$f_{W_1,w_2}(x) = w_2^T \sigma(W_1^T x)$$

where $x \in \mathbb{R}^m$, $W_1 \in \mathbb{R}^{m \times h}$, $w_2 \in \mathbb{R}^h$, and

$$\sigma(x) = 1/(1 + \exp(-x)).$$

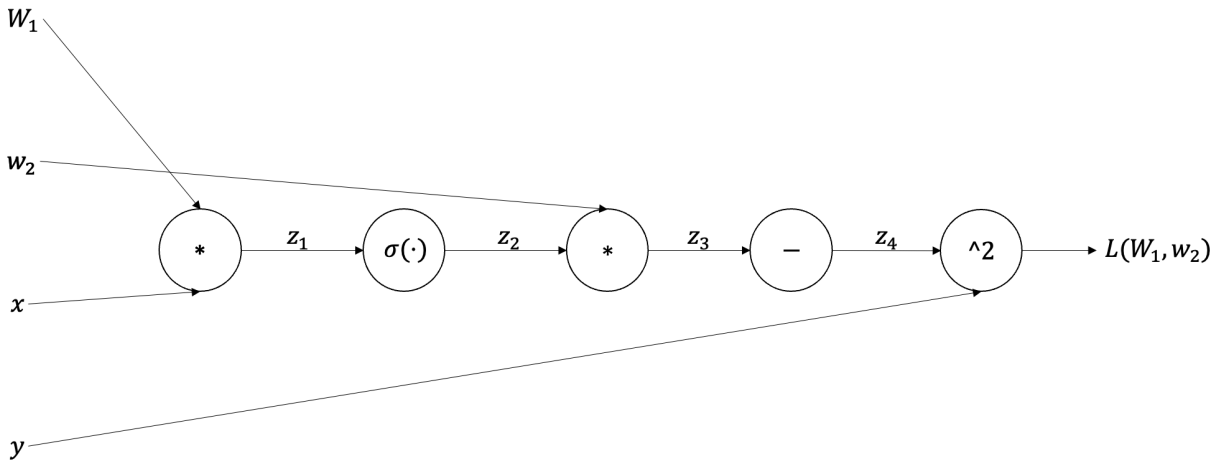
The subscript notation in f_{W_1,w_2} just indicates that W_1 and w_2 are parameters of the function f.

Suppose the neural network uses squared-error prediction loss for data points (x, y):

$$\mathcal{L}(y, f_{W_1,w_2}(x)) = (y - f_{W_1,w_2}(x))^2. \tag{1}$$

(a) For this loss function, draw the corresponding computation graph. (*Hint:* There are four intermediate nodes in the graph, which you should label $z_1, \ldots, z_4$.)

Solution: The computation graph for $\mathcal{L}(W_1, w_2)$ is as follows:



The trick to coming up with computation graphs is to "unroll" the expression into a series of operations on different quantities, starting from the innermost nested expression.

(b) Using the chain rule, write down an expression for $\frac{\partial \mathcal{L}(W_1,w_2)}{\partial w_2}$. Simplify your answer in terms of the intermediate nodes $z_1, \ldots, z_4$.

Solution: First note that $z_1 = W_1^T x$, $z_2 = \sigma(z_1)$, $z_3 = w_2^T z_2$, $z_4 = (y - z_3)$.

$$\begin{aligned}\frac{\partial \mathcal{L}(W_1, w_2)}{\partial w_2} &= 2(y - w_2^T \sigma(W_1^T x)) \cdot -\frac{\partial w_2^T \sigma(W_1^T x)}{\partial w_2} \\ &= 2(y - w_2^T \sigma(W_1^T x)) \cdot -\sigma(W_1^T x) \\ &= 2z_4 \cdot -z_2.\end{aligned}$$

(c) Using the chain rule, write down an expression for $\frac{\partial \mathcal{L}(W_1, w_2)}{\partial W_1}$. As in the previous part, simplify your answer in terms of the z_i as much as possible.

(*Hint 1:* In addition to the z_i, your answer will involve x, w_2, and the derivative of the sigmoid function.)

(*Hint 2:* If you are having trouble with the vector and matrix notation, start by assuming that w_2 and W_1 are both one-dimensional real numbers.)

Solution:

$$\begin{aligned}\frac{\partial \mathcal{L}(W_1, w_2)}{\partial W_1} &= 2(y - w_2^T \sigma(W_1^T x)) \cdot -\frac{\partial w_2^T \sigma(W_1^T x)}{\partial W_1} \\ &= 2z_4 \cdot x \left(-w_2 \odot \sigma'(W_1^T x)\right)^T \\ &= -2z_4 \cdot x \left(w_2 \odot \sigma'(z_1)\right)^T\end{aligned}$$

Here, $w_2 \odot z_2 \odot (1 - z_2)$ is the the element-wise product of three h-dimensional vectors, so it is also a h-dimensional vector. The cross-product of the m-dimensional vector x and the h-dimensional vector $w_2 \odot z_2 \odot (1 - z_2)$ is therefore a $m \times h$-dimensional matrix.

(d) Explain why simplifying the derivatives in terms of the z_i allows us to save computation.

Solution: It minimizes the number of variables we have to manipulate when differentiating.

Feedback Form

On a scale of 1-5, where 1 = much too slow and 5 = much too fast, how was the pace of the discussion section?

1 2 3 4 5

Which problem(s) did you find most useful?

Which were least useful?

Any other feedback?